Práctica 1
APPIV

Clasificación de imágenes. Arquitecturas CNN. Transfer Learning.

10 de abril de 2022

Gloria del Valle Cano
gloria.valle@estudiante.uam.es

1. Simple CNN

- Tamaños de los conjuntos de entrenamiento y validación descargados del dataset MNIST.

	Alto	Ancho	Nº de canales	Nº de muestras
Entrenamiento	28	28	1	60000
Validación	28	28	1	10000

Tabla 1: Resumen del tamaño de los conjuntos de *train* y *valid*.

- Número de parámetros del modelo Simple CNN.

	Número de parámetros del modelo Simple CNN
Simple CNN	813802

Tabla 2: Resumen del número de parámetros del modelo.

- Incluya las curvas de entrenamiento y validación para 10 épocas. Indique también la mejor precisión obtenida, y en qué época se logra este resultado. Comentar las conclusiones sobre la evolución de la loss de entrenamiento y validación, con respecto a posibles problemas de sesgo (high-bias) o sobreajuste (overfitting). Indique si considera que continuar con más épocas de entrenamiento mejoraría el rendimiento del modelo.

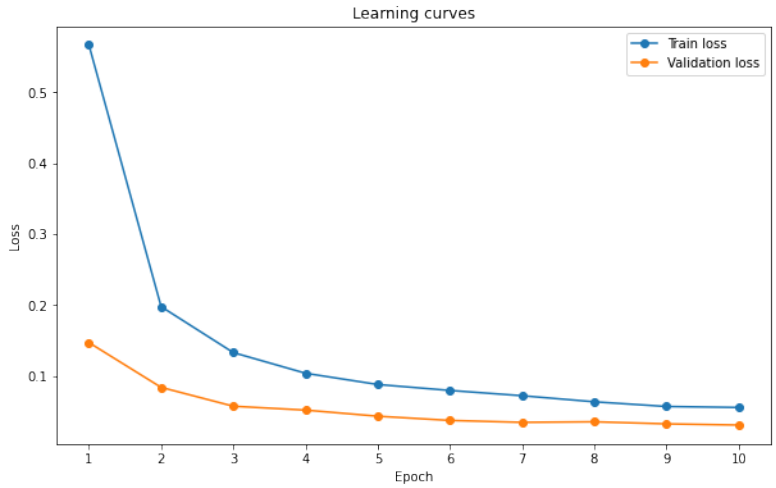


Figura 1: Curvas de entrenamiento y validación para el modelo Simple CNN. Épocas 1-10.

Como podemos ver en la Figura 1, la función de pérdida va decreciendo mucho más en el conjunto de validación que en el conjunto de entrenamiento, por lo que vemos que no hay *overfitting*. Es posible que este efecto se deba a la inclusión del *Dropout* en entrenamiento, ya que esto regulariza por la capacidad de ignorar determinadas neuronas durante el entrenamiento. Tampoco se observan problemas de *high-bias*, ya que la pérdida siempre decrece, por lo que podemos ver que el modelo está aprendiendo.

	Mejor precisión (validación)	Época con mejor precisión
Simple CNN	98.94 %	10

Tabla 3: Mejor precisión con su correspondiente época, obtenida por el modelo.

La mejor precisión obtenida se alcanza en la última época (ver Tabla 3), siendo ésta bastante alta. Vista la progresión de los resultados en cada época (ver primer notebook) y las curvas de aprendizaje, es muy posible que el modelo siguiera aprendiendo un poco más. Comprobamos esta suposición más tarde, con 10 épocas más. Adjuntamos los resultados en la Figura 2.

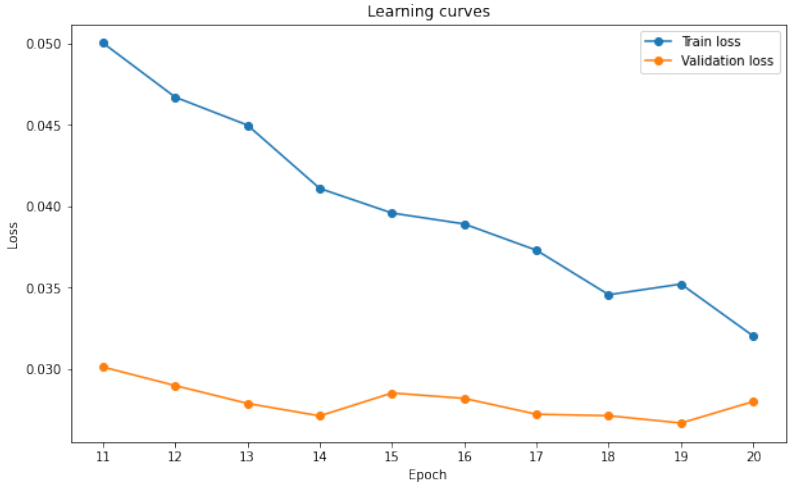


Figura 2: Curvas de entrenamiento y validación para el modelo Simple CNN. Épocas 11-20.

Viendo los resultados en el notebook, conseguimos aumentar la precisión a 99,10 % en la época 19. Observando el comportamiento de la curva y de los resultados, es muy posible que el aprendizaje se hubiera estancado en ese punto.

- Incluir la matriz de confusión obtenida. Dada esta matriz de confusión, informe de los 2 casos de confusión entre clases que ocurren con más frecuencia.

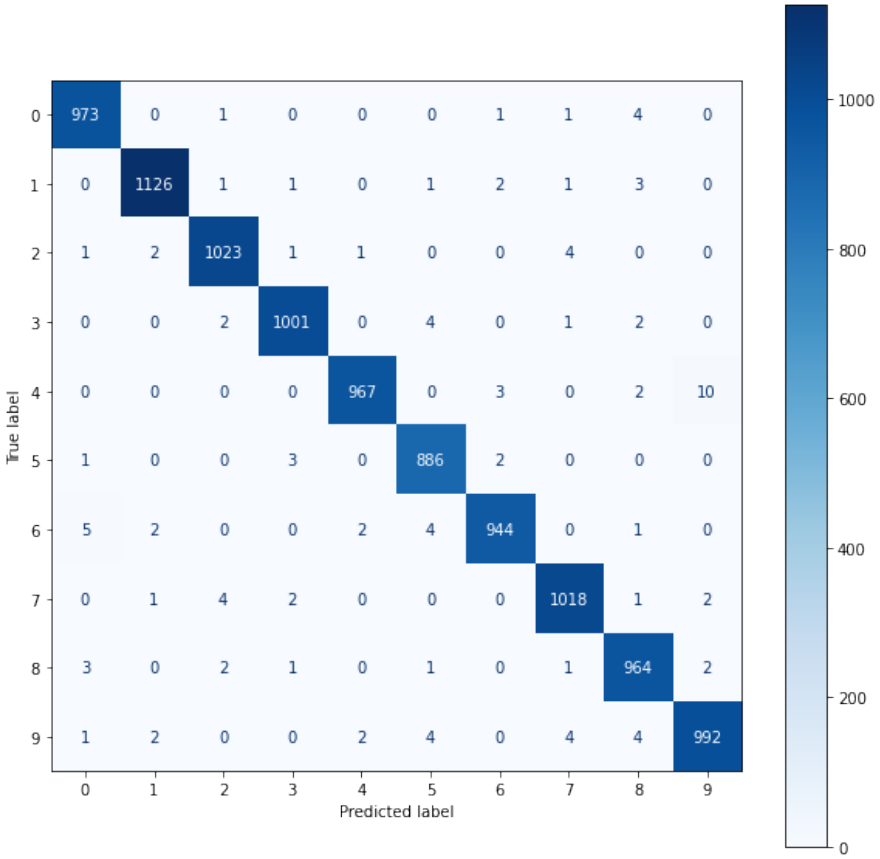


Figura 3: Matriz de confusión de Simple CNN (10 épocas).

Viendo los resultados, observamos que obtenemos un total de 106 errores de un total de 10000 predicciones. Los errores más comunes se encuentran en el dígito 6 (tiene muchas confusiones con el 0, el 5, etc.), el 4 (se suele confundir con el 9), el 5, incluso el 8. Podemos ver mejor algunas de estas confusiones en la Figura 4.

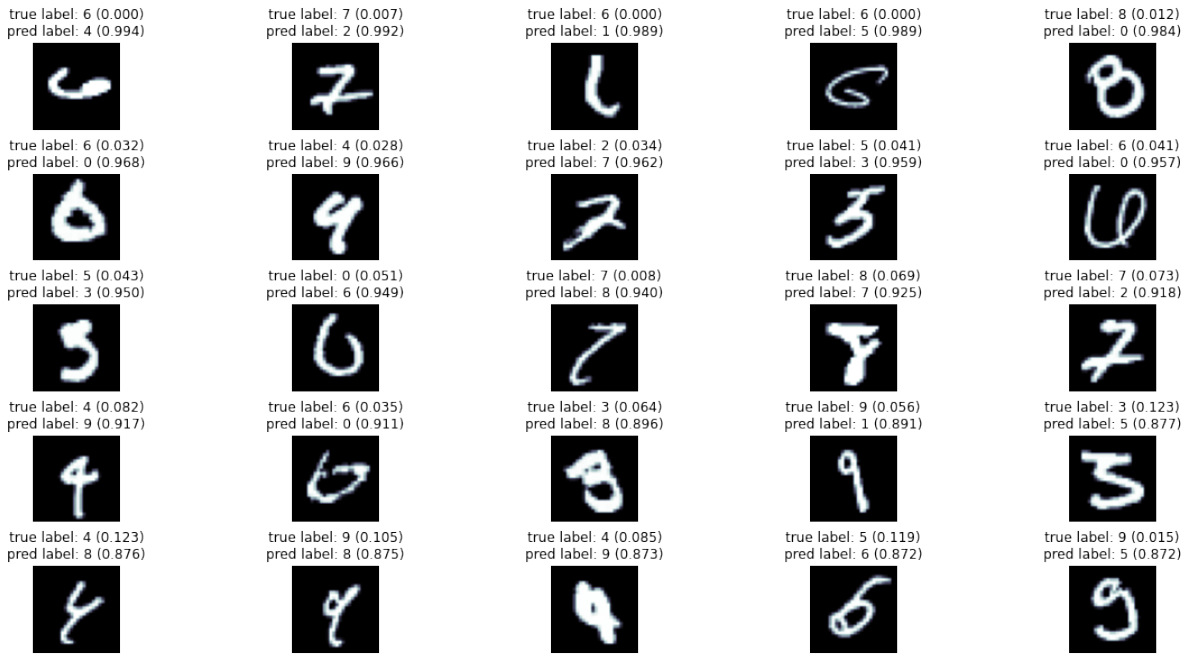
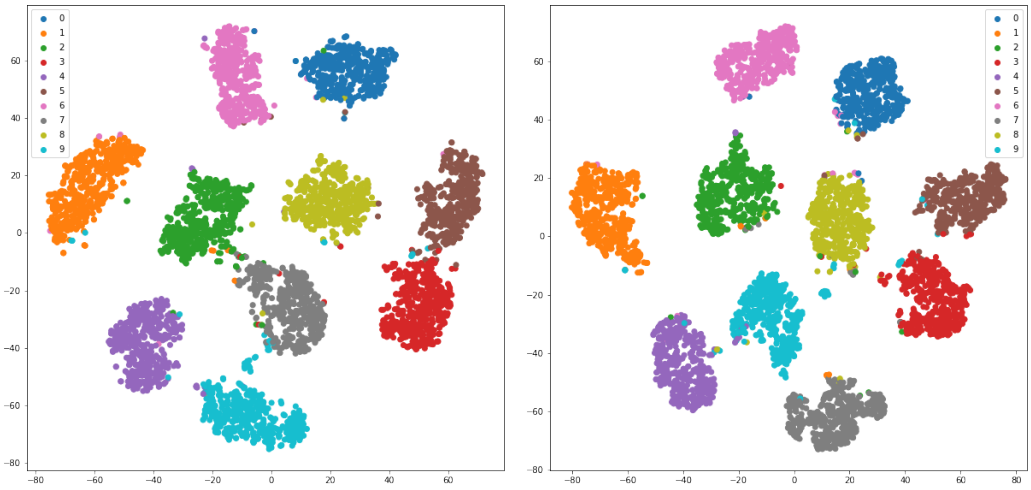


Figura 4: Ejemplos de dígitos con más errores en la clasificación.

- Comente las diferencias entre el gráfico t-SNE de la representación de las capas final e intermedia de la CNN, aplicado a las imágenes del conjunto de validación. Para ello, considere la proximidad y la dispersión entre los clústeres en ambas representaciones, y su relación con la capacidad de realizar una correcta clasificación de las muestras.



(a) Representación intermedia (b) Representación final

Figura 5: Representaciones t-SNE.

Como vemos en la Figura 5 obtenemos 10 clases completamente diferenciadas por clusters, por lo que parece que la red clasifica con éxito los dígitos aun con sus pocos fallos. Sí que es verdad que en la representación intermedia observamos que hay un poco de solapamiento en las clases 5 y 3 y en el 2 y el 7, si bien en la representación final vemos que este efecto se ha mitigado.

- Dadas las diferencias entre la representación t-SNE de ambas capas, y dada la arquitectura de la red implementada, identifique en qué capa de la red se extraen las características, y proponga una forma de reducir la complejidad de la red, con una penalización baja en la precisión de la clasificación.

  Observando la arquitectura de esta red vemos que las características se extraen antes de la primera capa *Linear*. Una manera de reducir la complejidad de la red sería eliminar la última capa lineal ya que parece que la diferencia entre ambas representaciones no es vital y el error no sería muy alto.

2. AlexNet

- Incluya el código que ha utilizado para definir la clase `AlexNet`.

```
    class AlexNet(nn.Module):
            def __init__(self, output_dim):
                super().__init__()

                self.features = nn.Sequential(
                    # First convolutional layer. Use 5x5 kernel instead of
                    ↪  11x11
                    nn.Conv2d(3, 48, 5, 2, 2), #in_channels, out_channels,
                    ↪  kernel_size, stride, padding
```

```
                    nn.MaxPool2d(2), #kernel_size
                    nn.ReLU(inplace = True),
                    # Complete the following four conv layers of the
                    ↪  AlexNet model.
                    # Subsampling is only performed by 2x2 max pooling
                    ↪  layers (not with stride in the
                    # convolutional layers)
                    # Pay special attention to the number of input and
                    ↪  output channels of each layer
                    nn.Conv2d(48, 128, 5, 1, 2),
                    nn.MaxPool2d(2),
                    nn.ReLU(inplace=True),

                    nn.Conv2d(128, 192, 3, 1, 1),
                    nn.ReLU(inplace=True),
                    nn.Conv2d(192, 192, 3, 1, 1),
                    nn.ReLU(inplace=True),

                    nn.Conv2d(192, 128, 3, 1, 1),
                    nn.MaxPool2d(2),
                    nn.ReLU(inplace=True)
                )

                self.classifier = nn.Sequential(
                    # First linear layer
                    nn.Dropout(0.5),
                    nn.Linear(128 * 2 * 2, 2048), # final conv layer
                    ↪  resolution 2x2
                    nn.ReLU(inplace = True),
                    # second linear layer
                    nn.Dropout(0.5),
                    nn.Linear(2048, 2048),
                    nn.ReLU(inplace=True),

                    # Last Linear layer. No ReLU
                    nn.Linear(2048, output_dim),
                )

            def forward(self, x):
                x = self.features(x)
                interm_features = x.view(x.shape[0], -1)
                x = self.classifier(interm_features)
                return x, interm_features
```

- Número de parámetros del modelo `AlexNet`.

  Con `summary` de `torchsummary` podemos ver este resultado.

	Nº parámetros entrenables
AlexNet	6 199 498

Tabla 4: Resumen del número de parámetros entrenables del modelo.

- Incluya las curvas de entrenamiento y validación para 15 épocas. Indique también la mejor precisión obtenida, y en qué época se logra este resultado. Comentar las conclusiones sobre la evolución de la loss de entrenamiento y validación, y comentar lo que posiblemente está sucediendo después de la época 10. Indique si considera que continuar con más épocas de entrenamiento mejoraría el rendimiento del modelo.

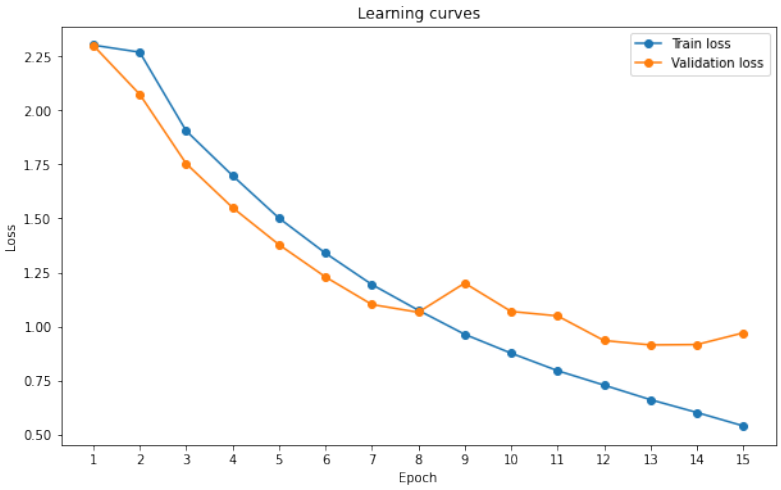


Figura 6: Curvas de entrenamiento y validación para el modelo AlexNet. Épocas 1-15.

Las curvas de entrenamiento y validación se pueden ver en la Figura 6. Vemos que esta vez el dataset es mucho más difícil de aprender y utilizando un modelo como AlexNet obtenemos la precisión que se observa en la Tabla 5. También observamos que la pérdida va descenciendo como se espera, si bien a partir de la época 8 la loss de validación empieza a aumentar, por lo que el modelo está empezando a sobreajustar. Vemos que entre 8 y, tal vez, 12 épocas sería suficiente y que muchas más no sería beneficioso para el rendimiento del modelo.

	Mejor precisión (validación)	Época con mejor precisión
AlexNet	71.22 %	14

Tabla 5: Mejor precisión con su correspondiente época, obtenida por el modelo.

- Incluir la matriz de confusión. Comentar los resultados obtenidos atendiendo a las características de las imágenes de cada clase.

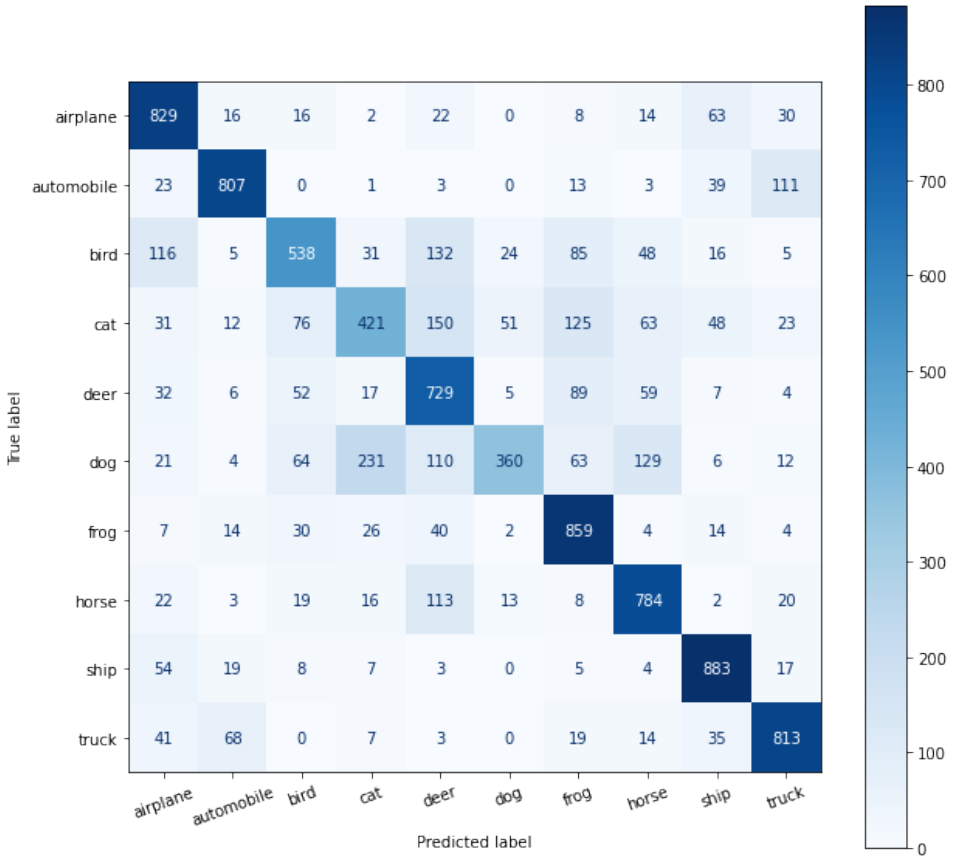


Figura 7: Matriz de confusión de AlexNet (15 épocas).

Según vemos en la Figura 7, las clases que obtienen más errores son las de *dog*, *cat* (se confunden entre sí bastante) y *horse*, que se suele confundir con *deer*. Esto puede tener sentido porque pueden ser parecidos entre sí por tamaño, color e incluso por los fondos, ya que los últimos se encuentran más en la naturaleza, mientras que los gatos y los perros son domésticos. También observamos confusiones entre *automobile* y *truck*, lo que parece igual de lógico. Podemos ver ejemplos de esto en la Figura 8.

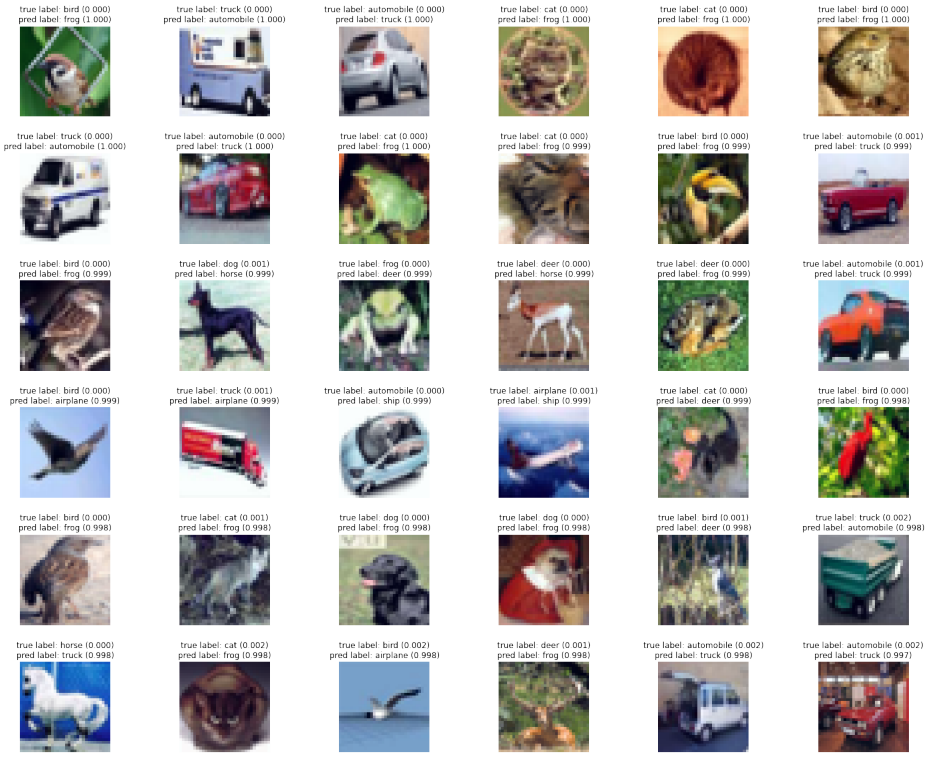


Figura 8: Ejemplos de clases con más errores en la clasificación.

- Incluya los resultados t-SNE para la capa última capa de la red: analice estos resultados (proximidad, dispersión, agrupación de clústeres) teniendo en cuenta la apariencia de las imágenes de las diferentes clases, sus características típicas y compare los resultados con los resultados t-SNE en el dataset MNIST.

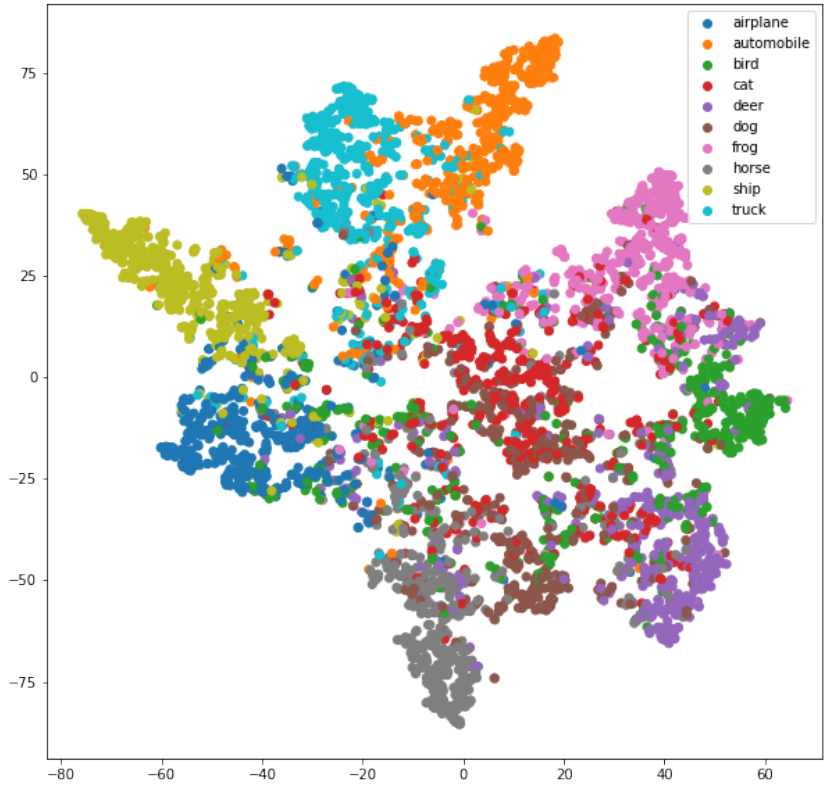


Figura 9: Representación final de AlexNet con la proyección t-SNE.

En la Figura 9 vemos que no se observan clusters tan diferenciados como en el anterior dataset con la otra red, si bien se pueden diferenciar igualmente la mayor parte de las regiones. Comprobamos que los errores han ocurrido en la mayor parte en las clases que hemos mencionado antes, además del caso de los pájaros. Asimismo, vemos que este caso no es tan bueno y que requiere de una mejora. Las maneras de mejorar esto son diversas, entre las cuales se encuentra una mejor optimización de hiperparámetros, en concreto cambiar el *learning rate* o el tamaño de *batch*. También podríamos realizar técnicas de *data augmentation* para aumentar y derivar de los ejemplos que se tienen del dataset, con el fin de enriquecer mucho más al modelo, si bien esto pueda tener un efecto contraproducente en el tiempo de entrenamiento.

3. Transfer Learning

- Precisiones obtenidas para las diferentes alternativas analizadas.

	Desde cero	Pre-entrenamiento + SVM	Ajuste fino	Ajuste fino + DA
Precisión	72.00 %	90.50 %	93.50 %	95.00 %

Tabla 6: Resultados obtenidos con aprendizaje por transferencia con el dataset Alien.

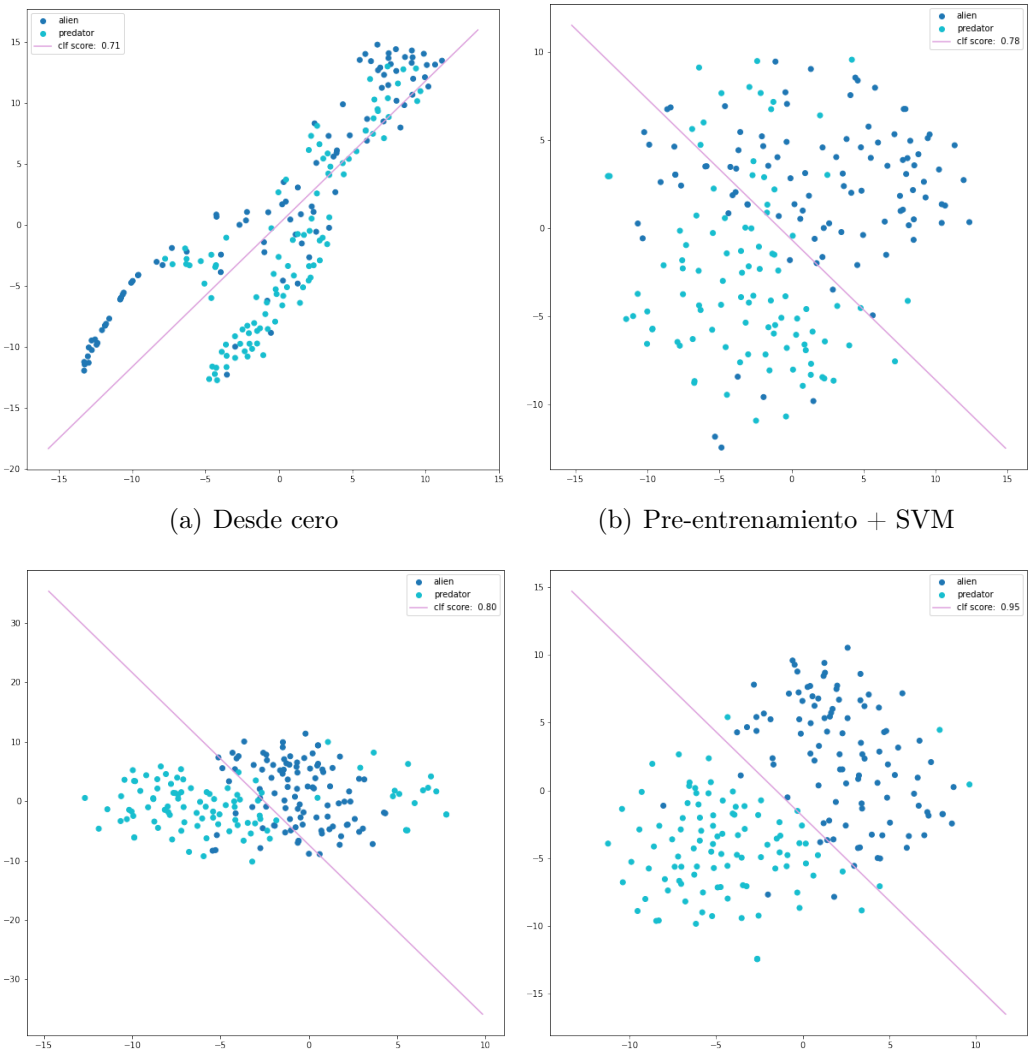
- Compare las representaciones t-SNE de las diferentes alternativas: entrenamiento desde cero, pre-entrenamiento + SVM, ajuste fino (sin data augmentation) y ajuste fino (con data augmentation) A partir de las diferentes representaciones obtenidas, en las cuatro alternativas analizadas., comente sus diferencias en cuanto a la capacidad de separar linealmente ambas clases, y el nivel de muestras clasificadas erróneamente dada esta separación lineal.

  En la Figura 10 se muestran las proyecciones t-SNE obtenidas para cada uno de los métodos, con un hiperplano que separa ambas clases.

  Realizamos las siguientes observaciones para cada uno de los métodos:

  - Desde cero: las clases no están muy separadas a diferencia de otros métodos.
  - Pre-entrenamiento + SVM: vemos que hay una diferencia mucho más notable de la separación de clases, aún con varios errores.
  - Ajuste fino: parece que mejora bastante más la clasificación.
  - Ajuste fino: comprobamos que el efecto de data augmentation es considerablemente bueno para estos datos y esta red. Existen pocos errores y la separabilidad es clara.

  Parece que data augmentation es una buena técnica de mejora de los resultados, si bien hay que controlar un poco la cantidad de datos nuevos que generamos y perfeccionar las técnicas utilizadas de cara a otros datasets. No sabemos si esto es generalizable para cualquier dataset, pero desde luego que parece una técnica prometedora para la mejora del rendimiento de estas redes en clasificación de imágenes.



(a) Desde cero

(b) Pre-entrenamiento + SVM

(c) Ajuste fino

(d) Ajuste fino + Data Augmentation

Figura 10: Representaciones t-SNE para cada uno de los métodos probados.